Application Note

AN_291

FT800_Create_Multi-Language_Font

Version 1.0

Issue Date: 2014-02-17

This application note describes how to display multi-language fonts on FT800 using an external character IC.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777   Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This application note describes how to display multi-language fonts on FT800 using an external character IC. Essentially the character IC provides raw data to allow a bitmap to be created for the required symbol that the FT800 can display.

1.1 Scope

This document shows how to display multi-language fonts on FT800 using an external character IC. The character IC - GT32L24A180 was selected for this example introduction. Example code is shown to explain how to output the multi-language characters on a display.

1.2 Overview

1.2.1 Hardware

The diagram below illustrates the overall hardware setup. The character IC and the FT800 are both SPI peripherals with independent chip select lines.

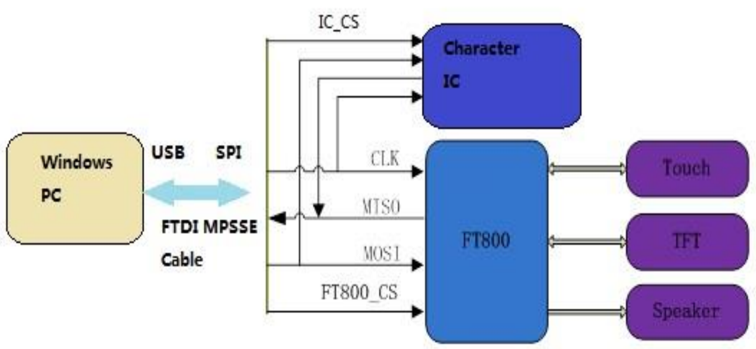

Figure 1-1: Block Diagram of Setup

1.2.2 Application Flow

The diagram below gives the basic flow to show how to display multi-language fonts on FT800.

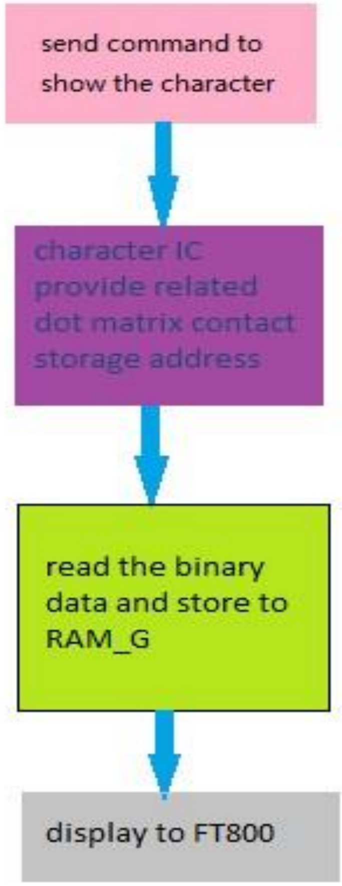

Figure 1-2: Application Flow

1.3 Software Required

- Visual Studio Express 2010 C++, can be downloaded from Microsoft's website.
  http://www.microsoft.com/visualstudio/cht/downloads#d-2010-express
- FTDI D2XX driver, can be downloaded from FTDI's website.
  http://www.ftdichip.com/Drivers/D2XX.htm

1.4 Hardware Required

- PC with Windows OS installed
- FT800 development module with LCD panel; VM800B or VM800C
- FTDI's MPSSE cable; C232HM-DDHSL-0
- Character IC; GT32L24A180

2 Introduction the Character IC GT32L24A180

2.1 General

GT32L24A180 contains a 15X16 dot matrix font size, supporting character sets including GB2312 /Unicode simplified Chinese, BIG5/Unicode traditional Chinese, JIS0208/Unicode Japanese, KSC5601 Korean and 173 countries characters in Unicode. Also supported is the Chinese PINYIN input method from a PINYIN code list.

The data format is arranged as vertical - byte, horizontal - string. The user may obtain the address of certain characters dot matrix with the calculation method given by this application note, which enables the user to access more character data by continually reading from the address already obtained.

2.2 Operation Instruction

2.2.1 Instruction Parameter

Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
Read	Read Data Bytes	0000 0001	03h	3	—	1 to ∞
Fast Read	Read Data Bytes at Higher Speed	0000 1011	0Bh	3	1	1 to ∞
WREN	Write Enable	0000 0110	06h	—	—	—
WRDI	Write Disable	0000 0100	04h			
PP	Page Program	0000 0010	02h	3	—	1 to 256
SE	Sector Erase	0010 0000	20h	3	—	—
BE	Block Erase(64K)	1101 1000	D8h	3	—	—
CE	Chip Erase	0110 0000/ 1100 0111	60h/ C7h	—	—	—

Table 1 Instruction Parameter

2.2.2 Character Dot Matrix Contact Storage Address

NO.	Font Content	Character Set	Number of Characters	Base Address
1	24*24 dot matrix GB2312 Font	GB2312	6763+282	0x100000
2	12*12 dot matrix GB2312 extend Font		32	0x17BED0
3	5*7 dot ASCII Font		96	0x17C4D0
4	7*8 dot ASCII Font		96	0x17C7D0
5	6*12 dot ASCII Font		96	0x17CAD0
6	8*16 dot ASCII Font		96	0x17D0D0
7	8*16 dot bold-faced ASCII Font		96	0x17D8D0

8	9*24 dot ASCII Font volume	ASCII Font Volume	96	0x1BC4D0
9	12 dot matrix proportional adjusted ASCII Arial Font		96	0x17DED0
10	12 dot matrix proportional adjusted ASCII Times New Roman Font		96	0x17E890
11	16 dot matrix proportional adjusted ASCII Arial Font		96	0x17F250
12	16 dot matrix proportional adjusted ASCII Times New Roman Font		96	0x17FF10
13	24 dot matrix proportional adjusted ASCII Arial Font		96	0x180BD0
14	32 dot matrix proportional adjusted ASCII Arial Font		96	0x1BD6D0
15	32 dot matrix proportional adjusted ASCII Times New Roman Font		96	0x1C0790
16	8*16 Latin Font	Unicode	496	0x182790
17	12*24 Latin Font		496	0x186890
18	12*24 Latin Arial Font		496	0x1C6F96
19	12 dot proportional adjusted Latin Font		496	0x19B550
20	16 dot proportional adjusted Latin Font		496	0x194B10
21	8*16 Greek Font		96	0x184690
22	12*24 Greek Font		96	0x18C590
23	12 dot proportional adjusted Greek Font		96	0x19E7B0
24	16 dot proportional adjusted Greek Font		96	0x198CF0
25	8*16 Cyrillic Font		208	0x184C90
26	12*24 Cyrillic Font		208	0x18D790
27	12*24 Cyrillic Arial Font		208	0x1CAB96
28	12 dot proportional adjusted Cyrillic Font		208	0x19F170
29	16 dot proportional adjusted Cyrillic Font		208	0x1999B0
30	8*16 Hebrew Font		112	0x185990
31	12*24 Hebrew Font		112	0x1CD296
32	8*16 Thai Font		128	0x186090
33	24 dot proportional adjusted Thai Font		128	0x1CE556
34	8*16 Japanese Font		64	0x1CF71C
35	12*24 Japanese Font		64	0x1CFB0C
36	16 dot proportional adjusted Arabic Font		576	0x18FE90
37	24 dot proportional adjusted Arabic Font		576	0x1B1090
38	8*16ISO8859NO.1～NO.16 (Without No.6 and No.12)	ISO 8859	128X14	0x1A0690
39	5*7ISO8859NO.1～NO.16 (Without No.6 and No.12)		128X14	0x1A7690

40	5*10 LCM Font	LCM	256	0x1AAE90
41	5*7 LCM Font		256	0x1AF490
42	LCM Font Spare Area-1(5*10)		256	0x1AFC90
43	LCM Font Spare Area-2(5*10)		256	0x1B0690
44	437--USA, Standard Europe)	CODE PAGE	256	0x1D82CC
45	737--Greek		256	0x1D84CC
46	775--Baltic		256	0x1D86CC
47	850--Multilingual		256	0x1D88CC
48	852--Latin 2		256	0x1D8ACC
49	855--Cyrillic		256	0x1D8CCC
50	857--Turkish		256	0x1D8ECC
51	858--Euro		128	0x1D90CC
52	860--Portuguese		256	0x1D91CC
53	862--Hebrew		256	0x1D93CC
54	863--Canadian French		256	0x1D95CC
55	864--Arabic		256	0x1D97CC
56	865--Nordic		256	0x1D99CC
57	866--Cyrillic 2		256	0x1D9BCC
58	1251--Cyrillic		256	0x1D9DCC
59	1252--Latin Ⅰ		256	0x1D9FCC
60	1253--Greek		256	0x1DA1CC
61	1254--Turkish		256	0x1DA3CC
62	1255--Hebrew New		256	0x1DA5CC
63	1256--Arabic		256	0x1DA7CC
64	1257--Baltic		256	0x1DA9CC
65	928--Greek		96	0x1DABCC
66	Hebrew old code		96	0x1DAC8C
67	International character code		132	0x1DAD4C
68	24 dot matrix katakana Font		64	0x1D070C
69	14*28 Blackbody half angle_numerical symbols	Numerical Symbols	15	0x1C3850
70	20*40 Blackbody half angle_numerical symbols		12	0x1C3B98
71	28 dot proportional adjusted numerical symbols		15	0x1C4138
72	40 dot proportional adjusted numerical symbols		13	0x1C47E6
73	Bar Code numerical symbols_EAN13	Bar Code	60	0x1C515E
74	Bar Code numerical symbols_CODE128		107	0x1C5E06

75	Antenna Symbol	Other Symbol	5	0x1C6EBE
76	Battery Symbol		4	0x1C6F36

Table 2 Character Dot Matrix Contact Storage Address

2.2.3 Calculation of Character Address

GB2312 24*24 dot matrix GB2312 Font was selected for this example.

2.2.3.1 Parameter Declaration

GBCode indicates Chinese characters code.

MSB indicates GBCode's high 8 bits.

LSB indicates GBCode's low 8 bits.

Address indicates Chinese characters or ASCII character dot matrix address byte in the chip.

2.2.3.2 Calculation Method

```
HZ2424ZF_ADDR = 0X100000;
HZ2424HZ_ADDR = 0X104F50;
// Full width symbols, used to support the character，282 Bytes
if(MSB >=0xA1 && MSB <= 0xA3 && LSB >=0xa1 )
{
temp = (MSB - 0xA1) * 94 + (LSB - 0xA1);
Address = temp *72 + HZ2424ZF_ADDR;
}
// Chinese characters 6763 Bytes
else if(MSB >=0xb0 && MSB <= 0xf7 && LSB >=0xa1)
 {
temp= (MSB - 0xB0) * 94 + (LSB - 0xA1);
Address = temp*72 + HZ2424HZ_ADDR;
 }
```

3 Example Application Code

For the user's easy understanding, this example application will show Chinese character "高" on FT800, using FT800_SampleApp_1.0 as base.

a. Get the input character's coding. Using GB2312 as example, with the Chinese character "高" in GB2312 code, its HEX code is MSB: 24+0xA0, LSB: 63+0xA0

b. The character IC provides its own calculation between character Hex value and the chip's related dot matrix contact storage address, with the IC GT32L24A180, the address in the chip is: 0x113440

c. Following the GT32L24A180 reading instruction, read the bytes out from the chip, the length should be: 24x24(3bytes*24=72bytes)

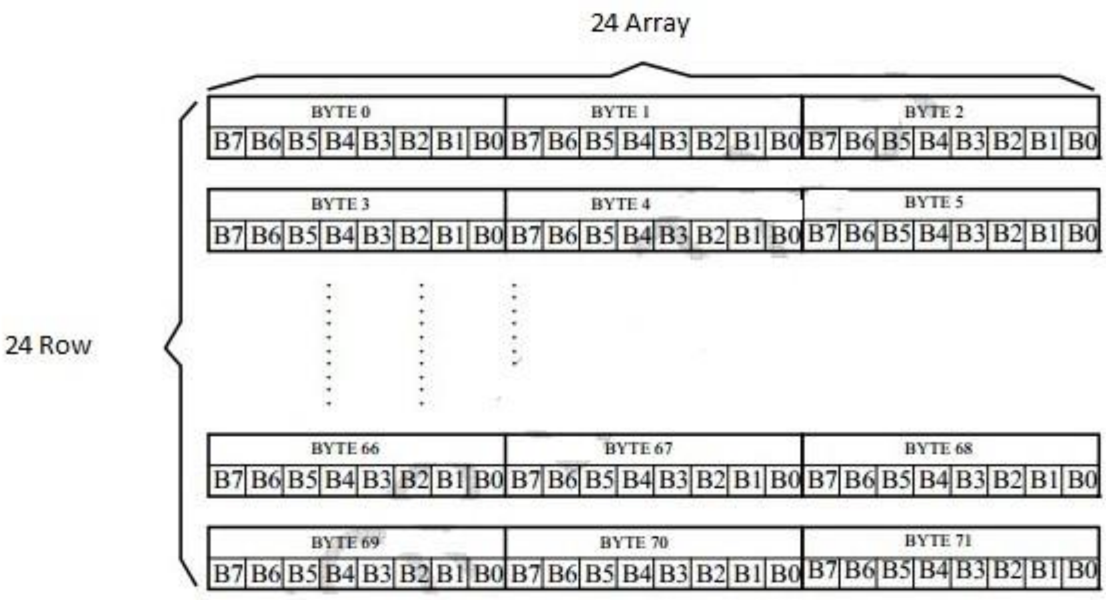

Figure 3-1: 24*24 dot Chinese characters arrangement

d. The content should be displayed as a bitmap in the FT800 in L1 format:

L1 format layout		Alignment
Pixel 0	Bit 7	Byte 0
Pixel 1	Bit 6	
....		
Pixel 7	Bit 0	

Figure 3-2: L1 format layout

Each ON ('1') bit represents a characters related color dot, each off ('0') bit represents the background color, the other parts are don't care.

e. Put the "高" dot matrix contents in Src/SampleApp_RawData.c (The array mechanism is chosen for easy teaching, a user may choose alternative coding styles e.g. read binary from file, reading from flash program space, and store the binary data into a buffer malloced, then save to RAM_G when displayimg).

- unsigned char gao[] = {0, 0x30, 0, 0, 0x18, 0, 0, 0x10, 0x0c, 0x7f, 0xff, 0xfe, 0, 0, 0, 0x02, 0x01, 0x80, 0x03, 0xff, 0xc0, 0x03, 0x01, 0x80, 0x03, 0x01, 0x80, 0x03, 0xff, 0x80, 0x02, 0x01, 0, 0x20, 0, 0x18, 0x3f, 0xff, 0xfc, 0x30, 0, 0x18, 0x31, 0x03, 0x18, 0x31, 0xff, 0x98, 0x31, 0x83, 0x18, 0x31, 0x83, 0x18, 0x31, 0xff, 0x18, 0x31, 0x82, 0x18, 0x31, 0, 0x18, 0x30, 0, 0xf8, 0x30, 0, 0x30, 0x20, 0, 0x20};

f. Add the bitmap header information in SAMAPP_Bitmap_RawData_Header:

- SAMAPP_Bitmap_header_t SAMAPP_Bitmap_RawData_Header[] =

```
{
        /* format,width,height,stride,arrayoffset */
        {RGB565,     40,    40,    40*2,  0     },
        {PALETTED,   40,    40,    40,    0     },
        {PALETTED,   480,   272,   480,   0     },
        {L1, 24, 24, 3, 0},
};
```

g. Modify SAMAPP_GPU_BitmapFont() in Src/SampleApp.c:

- Create an extern definition of gao[] in SampleApp.c: extern unsigned char gao[];
- Change the SAMAPP_Bitmap_RawData_Header index to 3;
- Replace the "SAMAPP_Bitmap_RawData" in SAMAPP_GPU_Bitmap() by "gao"

Reference sample code below:

```
ft_void_t SAMAPP_GPU_BitmapFont()
{
  SAMAPP_Bitmap_header_t *p_bmhdr;
  ft_int16_t BMoffsetx,BMoffsety;

  p_bmhdr = (SAMAPP_Bitmap_header_t *)&SAMAPP_Bitmap_RawData_Header[3];
  /* Copy raw data into address 0 followed by generation of bitmap */
  Ft_Gpu_Hal_WrMemFromFlash(phost, RAM_G,&gao[p_bmhdr->Arrayoffset], p_bmhdr->Stride*p_bmhdr->Height);

  Ft_App_WrDlCmd_Buffer(phost, CLEAR(1, 1, 1)); // clear screen
  Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(255,255,255));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SOURCE(RAM_G));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_LAYOUT(p_bmhdr->Format, p_bmhdr->Stride, p_bmhdr->Height));
  Ft_App_WrDlCmd_Buffer(phost,BITMAP_SIZE(NEAREST, BORDER, BORDER, p_bmhdr->Width, p_bmhdr->Height));
  Ft_App_WrDlCmd_Buffer(phost,BEGIN(BITMAPS)); // start drawing bitmaps
  BMoffsetx = ((FT_DispWidth/4) - (p_bmhdr->Width/2));
  BMoffsety = ((FT_DispHeight/2) - (p_bmhdr->Height/2));
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2II(BMoffsetx, BMoffsety, 0, 0));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(255, 64, 64)); // red at (200, 120)
  BMoffsetx = ((FT_DispWidth*2/4) - (p_bmhdr->Width/2));
  BMoffsety = ((FT_DispHeight/2) - (p_bmhdr->Height/2));
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2II(BMoffsetx, BMoffsety, 0, 0));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(64, 180, 64)); // green at (216, 136)
  BMoffsetx += (p_bmhdr->Width/2);
  BMoffsety += (p_bmhdr->Height/2);
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2II(BMoffsetx, BMoffsety, 0, 0));
 Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(255, 255, 64)); // transparent yellow at (232, 152)
  Ft_App_WrDlCmd_Buffer(phost,COLOR_A(150));
  BMoffsetx += (p_bmhdr->Width/2);
  BMoffsety += (p_bmhdr->Height/2);
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2II(BMoffsetx, BMoffsety, 0, 0));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_A(255));
  Ft_App_WrDlCmd_Buffer(phost,COLOR_RGB(255,255,255));
  Ft_App_WrDlCmd_Buffer(phost,VERTEX2F(-10*16, -10*16));//for -ve coordinates use
```

```
vertex2f instruction
  Ft_App_WrDlCmd_Buffer(phost,END());
  Ft_App_WrDlCmd_Buffer(phost, DISPLAY() );

  /* Download the DL into DL RAM */
  Ft_App_Flush_DL_Buffer(phost);

  /* Do a swap */
  SAMAPP_GPU_DLSwap(DLSWAP_FRAME);
      SAMAPP_ENABLE_DELAY();
}
```

h. Build the project and run. The Chinese character will be displayed via the FT800 as shown below:

Figure 3-3: sample code test result

4 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited (USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.support@ftdichip.com
E-Mail (General Enquiries) us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales) cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries) cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Appendix A – References

Document References

1. datasheet for VM800C: DS_VM800C EVE
2. datasheet for VM800B: DS_VM800B EVE
3. AN_240 FT800 From the Ground Up
4. FT800 Programmer Guide
5. FT800 Embedded Video Engine Datasheet
6. Sample Application
7. Datasheet for GT32L24A180

Acronyms and Abbreviations

Terms	Description
EVE	Embedded Video Engine
IDE	Integrated Development Environment
MPSSE	FTDI Multi-Protocol Synchronous Serial Engine
MSVC	Microsoft Visual Studio C++ 2010
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus
VM800B/C	VM800B or VM800C board

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_291 FT800_Create_Multi-Language_Font

Document Reference No.: FT_000971

Clearance No.: FTDI# 371

Product Page: http://www.ftdichip.com/EVE.htm

Document Feedback: Send Feedback

Revision	Changes	Date
1.0	First release	2014-02-17